SAM & MAX

search this site...

Programmation | Admin System | Philo et culture | Cul | Cours et tutos | Nos tweets SANS twitter :-) | Multiboards | Nos githuberies

Home » Administration System » Le déploiement par conteneurs avec Docker

Le déploiement par conteneurs avec Docker

Posted by Sam on 15 May 2014 in Administration System | 1 comment

Mettre son projet en production, c'est la galère. Tellement que mille méthodes on vu le jour pour automatiser tout ça. Chef, salt, fabric, des script bash, virtualenv, git hooks, etc.

Après, il y a ceux qui utilisent des VM. Qui ont leur propres outils d'automatisation type Vagrant.

Et comme ça suffit pas, des services ce sont mis en place pour faciliter la mise en prod dans le cloud comme heroku, gondor…

Malgré ça, Max fait encore beaucoup de truc à la main parce que "ça marche jamais comme prévu". Pas très scalable.

Dernièrement, grâce à notre cher **Cortex**, j'ai découvert un projet écrit en Go nommé **Docker**, qui propose encore une autre approche du problème.

J'ai été intrigué après avoir visionné une conf sur le sujet car le principe est très cool, mais aussi parce que j'ai bossé à une époque avec **un des mecs** de chez docker. Et ce gars est un monstre. Mais vraiment. Une brute. Le genre qui n'a pas besoin de souris ni de gestionnaire de fenêtre, mais qui peut vous régler votre problème en tapant d'une main tout en vous expliquant ce qu'il fait dans votre domaine d'expertise alors que ce n'est pas le sien. Je l'ai vu faire. C'est énervant.

Pour le moment, je dois dire que Docker est vraiment sympas. Petit tour du propriétaire.

C'est comme si chaque process avait sa mini VM

Pour faire simple, docker, c'est de la virtualisation légère.

Ça marche comme cela : on prend une image d'un Linux de base qu'on fait tourner dans docker, on lui installe de quoi faire tourner un process – par exemple redis – et on obtient une nouvelle image qui contient juste la différence avec l'image précédente. On fait ça avec plein d'autres process, et quand on a besoin d'un des process, on lance l'image qui contient l'install de ce process.

Ce sont des images très légères, on peut en lancer 100 sur une même machine si on le souhaite. Mais elles sont parfaitement isolées : elles ont leur propre système de fichier, leurs propres arbres de processus, utilisateurs, permissions, ports… Donc si par exemple je fais un nginx compilé avec des extensions exotiques et une config zarb, je peux en faire une image puis l'utiliser sur n'importe quel serveur qui contient docker.

Le but, c'est de créer plein de petits conteneurs légers et isolés de votre machine. Et au lieu de configurer chaque serveur, on envoie juste les conteneurs dont on a besoin, et on est certain qu'ils marchent partout pareil, peu importe la config du serveur. **On virtualise des services, qu'on combine, et non un serveur complet.**

Quand je vous dis que c'est léger, c'est vraiment léger. A peine plus gourmand que le process sans la VM. En fait, un conteneur docker démarrer presque qu'instantanément, il n'y a pas de "boot" comme pour une vraie VM. Mais on en a les avantages car votre redis

Une lecture plus fastoche ?

Follow us @sam_et_max

Souscrire à nos conneries

Entrez votre adresse mail et vous recevrez une notification à chaque nouvel article.

Rejoignez les 312 abonnés

françis@ça-mousse.com

J'adhère bon dieu!

Catégories

- Administration System (109)
- Cul (59)
- Gadget (5)
- Gadgets (2)
- NSFW (1)
- Philo et culture (157)
- Programmation (416)
- Uncategorized (6)
- Web (46)

Posts récents

- Le déploiement par conteneurs avec Docker
- Peut-on compiler un programme Python ?
- Le pandas c'est bon, mangez en
- ……

dockerisé marchera pareil sur une Fedora et une Ubuntu.

Docker ne marche que sur Linux pour le moment, et encore, sur un Linux pas trop vieux car il utilise une technologie dont vous avez probablement peu entendu parlé dernièrement : **LXC**.

Détail amusant : docker est pas mal utilisé dans la communauté des devs sous Mac, qui du coup ont une VM Ubuntu dans laquelle ils font tourner docker pour travailler. N'est-ce pas merveilleux ?

La démonstrations obligatoire

Je ne vais pas vous laisser comme ça et vous demander de vous la mettre derrière l'oreille, donc exemple d'utilisation sous Bubuntoune.

Je suis en 14.04, mais je pense que ça marche pareil depuis la 12.04.

Installation :

```
$ sudo apt-get install docker.io
```

Cela va mettre la commande `docker.io` à votre disposition, mais sous d'autres systèmes, elle s'appelle juste `docker`. Perso je fais un alias docker="sudo docker.io" puisque de toute façon il faut toujours lancer cette commande avec les droits admin.

Ensuite, on va télécharger une image de base sur laquelle baser toutes nos images :

```
$ docker pull ubuntu
```

Docker.io, ce n'est pas juste un software, c'est aussi un service qui héberge les images. Vous pouvez bien entendu créer vos propres images de base, et héberger votre propre dépôt d'images, mais on va pas se faire chier à faire ça ici. Prévoyez quelques gigas de place, docker ne prends pas beaucoup de ressources CPU/RAM, mais par contre ça bouffe en espace disque.

Donc là, je demande la récupération des images "ubuntu", qui vont nous servir d'images de base. Elles sont toutes faites et prêtes à l'emploie, que demande le peuple ?

Ça va downloader pendant pas mal de temps, puis vous aurez plusieurs images à votre disposition, que l'on peut lister ainsi :

```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VI
REPOSITORY          TAG                 ID                  CREATED             SI
ubuntu              12.04               8dbd9e392a96        4 months ago        13
ubuntu              12.10               b750fe79269d        5 months ago        24
ubuntu              latest              8dbd9e392a96        4 months ago        13
ubuntu              precise             8dbd9e392a96        4 months ago        13
ubuntu              quantal             b750fe79269d        5 months ago        24
```

Vous faites votre marché : quelle image de base voulez-vous utiliser ?

La 12.04 est une LTS qui est déjà bien field testée, donc je vais prendre celle là.

Ensuite, pour lancer une commande avec, il faut faire `docker run id_image commande`. Ceci va lancer l'image, lancer la commande, et dès que la commande se termine, arrêter l'image :

```
$ docker run 74fe38d11401 echo 'YEAHHHHHHHHHHHHHH'
WARNING: Local (127.0.0.1) DNS resolver found in resolv.conf and containers can't
YEAHHHHHHHHHHHHHH
```

Comme vous le voyez, démarrer un conteneur, c'est vraiment rapide. Mais là, ça ne sert pas à grand chose :)

Faisons quelque chose de plus utile : lançon un terminal et demandons un accès à stdin/stdout ainsi qu'un tty :

```
$ docker run -i -t 74fe38d11401 bash
WARNING: Local (127.0.0.1) DNS resolver found in resolv.conf and containers can't
root@8195e92a5e62:/#
```

Arg

- Quelle est la différence entre un bug et un virus ?
- Cet article est trop gay
- La martingale à la roulette en Python
- Input number et spin buttons

Posts Populaires

- On a un nouveau projet et on a besoin d'UNE info sur vous (262)
- Entreprendre en France, Autopsie d'un échec. (114)
- Un gros Troll de plus sur Javascript (101)
- FizzBuzz en Python (98)
- Beaucoup de femmes sont de mauvais coups (94)
- Une journée dans un FKK (maison close Allemande) (89)
- Fêtons le départ de Max (87)
- On est en 2013, on a marché sur la lune, mais on a toujours pas… (85)

TagCloud

bash bitcoin blog bottle cache comprehension-lists css cul decorator **django** email encoding error fun **git** html import ipython iterable **javascript** jquery lambda linux mac **meta** mysql nginx nsfw php pip poo **python** python 3 redis ruby server shell ssh sublime text twitter **ubuntu** unit tests unpacking virtualenv yield

All Tags

Archives

- May 2014 (7)
- April 2014 (10)
- March 2014 (23)
- February 2014 (27)
- January 2014 (17)
- December 2013 (27)
- November 2013 (25)
- October 2013 (27)
- September 2013 (1)
- August 2013 (31)

Et hop, comme le process ne se termine pas tant qu'on est connecté au tty, on a le conteneur qui continue de tourner, mais on a un terminal dessus, nous permettant d'entrer n'importe quelle commande.

Si on installait **0bin** ?

D'abord, on a besoin de pip dans le conteneur:

```
# apt-get install python-pip
```

Notez qu'on est toujours root dans son conteneur. Après tout, c'est virtuel et isolé, donc pas de raison de se faire chier.

Ensuite, on tape dans pypi :

```
# pip install zerobin
```

Pas besoin de virtualenv, on est déjà dans un environnement isolé.

Faisons un petit fichier de config. On va avoir besoin de vim :

```
# apt-get install vim
# cd /home
# vi settings.py
```

On met des settings perso pour zerobin, histoire de montrer le principe de faire un conteneur avec un setup sur mesure (on peut, bien entendu, faire 1000 fois plus compliqué, c'est un exemple bande de bananes) :

```
# on le fait ecouter sur l'exterieur
HOST = "0.0.0.0"
# on change le menu de la page d'accueil
MENU = (
    ('Home', '/'),
    ('Contact', 'mailto:mysupermail@awesomesite.com')
)
```

Et on lance notre petit zerobin :

```
# zerobin --settings-file=settings.py
```

Et voilà, on a un zerobin qui tourne dans notre conteneur isolé.

On va sauvegarder tout ça. Sur notre machine hôte (donc pas dans le conteneur), on va sauvegarder notre nouvelle image. D'abord, on va trouver l'ID du conteneur qui tourne :

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STA
8195e92a5e62        ubuntu:12.04        bash                37 minutes ago      Up
```

Ensuite on commit :

```
$ docker commit 8195e92a5e62 zerobin
```

On peut tranquillement arrêter notre conteneur :

```
$ docker stop 8195e92a5e62
```

Maintenant on peut pusher notre image sur un repo distant avec `docker push` ou juste l'exporter en tar.gz et l'importer avec `docker export/import` et le mettre sur un serveur. Je ne vais pas le faire ici car c'est long et je pense que vous pouvez trouver **comment faire** sans moi.

Dans tous les cas, une fois sur le serveur, vous pouvez relancer votre image docker :

```
# docker run -t -i -p 7777:8000 zerobin bash
```

Cette fois, on rajoute une option de plus : `-p 7777:8000` dit à docker de relier le port `7777` de ma machine au port `8000` de mon conteneur. Ainsi, si une app va sur le port `7777` de ma machine, elle va en fait parler au port `8000` du conteneur sans s'en rendre compte.

Du coup, je peut lancer mon petit zerobin depuis mon contenur :

- July 2013 (32)
- June 2013 (24)
- May 2013 (30)
- April 2013 (31)
- March 2013 (33)
- February 2013 (29)
- January 2013 (34)
- December 2012 (32)
- November 2012 (36)
- October 2012 (35)
- September 2012 (36)
- August 2012 (37)
- July 2012 (33)
- June 2012 (24)
- May 2012 (31)
- April 2012 (20)
- March 2012 (10)
- February 2012 (14)

Flux RSS

RSS - Posts

Planet Python fr

Sélection d'articles sur Python du Web français

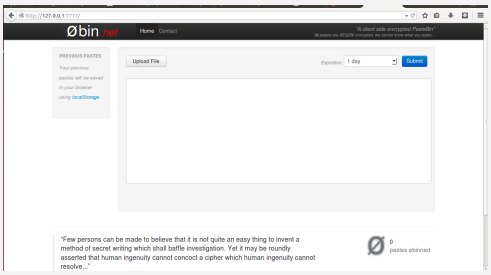
Envoyez des sioux

On adooooore les bitcoins :

19zAHPPuce4BAhsdy9KaFwVLurEJXMhMAn

```
# cd /home
# zerobin --settings-file=settings.py
```

Et voilà ! J'ai un zerobin qui marche, qui est préconfiguré, isolé et portable. Si je change de serveur, ou même d'OS, je peux juste reprendre le même conteneur et le relancer tel quel. Et je peux faire ça avec plein de process et les faire communiquer entre eux.



J'accède à 7777 sur la machine, mais ça tape sur 8000 dans mon conteneur

Docker est un outil très riche, et vous vous doutez bien que je ne vous ai montré que le début.

Par exemple, vous voulez sans doute ne pas avoir à lancer bash, puis un `cd`, puis zerobin. Ça peut s'automatiser avec un **dockerfile**. Vous voulez aussi peut être lancer le conteneur en arrière plan, ça se fait avec l'option `-d`. Ou peut être voulez-vous un dossier partagé entre tous les conteneurs ? C'est possible avec l'option **volume**.

Parmi les usages vraiment intéressants de dockers : packager les services très custos comme les nginx ou ffpeg compilés avec des options cryptiques, deployer son app django avec toutes les libs Python et les settings qui vont bien, remplacer une app à chaud en redirigeant juste le load balancer sur le nouveau conteneur, ou encore fournir un env de test à un client.

Je vous laisse prendre en main le nouveau joujou, j'ai des serveurs à planter.

Partager: Email More

No related posts.

Flattr this!

One comment

k3c
15/05/2014

Merci pour ce tuto, ça va me mettre le pied l'étrier.

Typo je suppose
s/ffpeg/ffmpeg

Flux RSS des commentaires

Leave a Reply

Your email address will not be published. Required fields are marked *

Name *

Email *

Website

Comment

You may use these HTML tags and attributes: `<a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code> <del datetime=""> <em> <i> <q cite=""> <strike> <strong> <pre lang="" line="" escaped="" highlight="">`

Submit Comment

Notify me of follow-up comments by email.

Notify me of new posts by email.

Jouer à mario en attendant que les autres répondent

contactez-nous

Designed by Elegant WordPress Themes